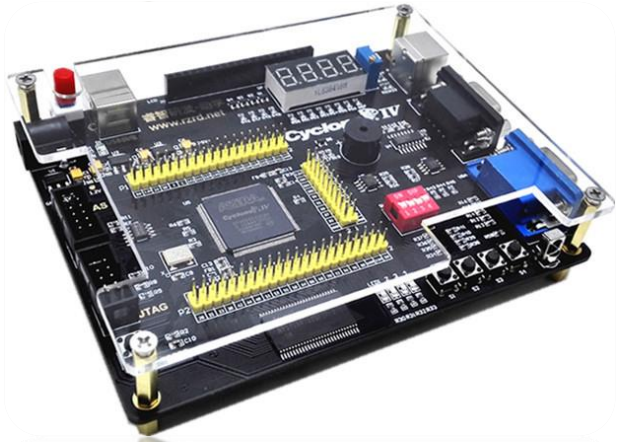
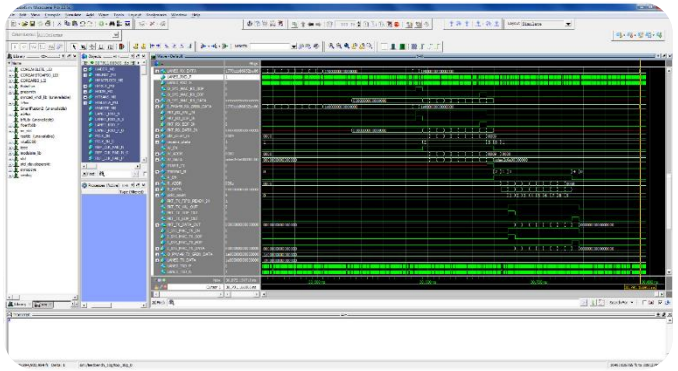
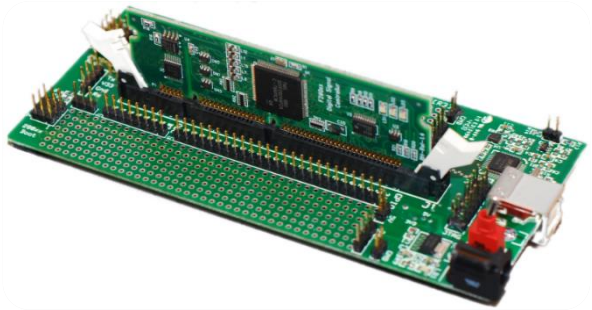
Cours FPGA, VHDL et DSP

Master d'Université Spécialisé en Ingénierie des Systèmes Embarqués (ISE)







Année universitaire 2024/2025

Dr. Ing. LASSIOUI Abdellah

Chapitre 4

Objectifs du chapitre N4

- ✓ Comprendre l'utilité de la méthodologie top-down utilisée pour la conception des circuits numériques
- ✓ Comprendre la notion de module en VHDL
- ✓ Comprendre comment instancier des modules dans un fichier VHDL
- ✓ Appliquer la méthodologie top down pour la conception d'une unité arithmétique et logique UAL

Flot de conception d’un système électronique

La méthodologie Top-down

La méthodologie Top-Down est une méthode de conception des systèmes électronique permettant d’avoir une compréhension détaillée et complète au niveau du système avant sa mise en œuvre. Elle consiste à **décomposer des systèmes compliqués en des simples tâches à réalisées.**

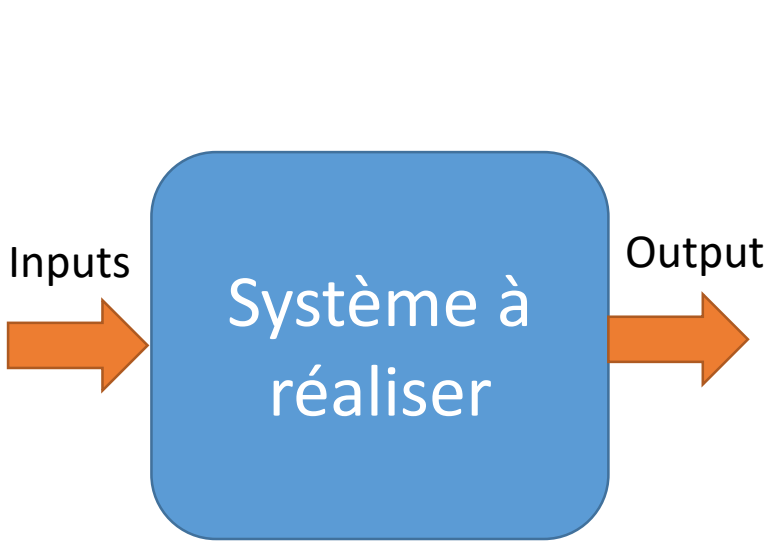
1ère Phase: On modélise le système sous forme de boite noire. "Je sais ce que ça fait, mais pour le moment je ne sais pas comment.«
On définit les entrées et les sorties du système

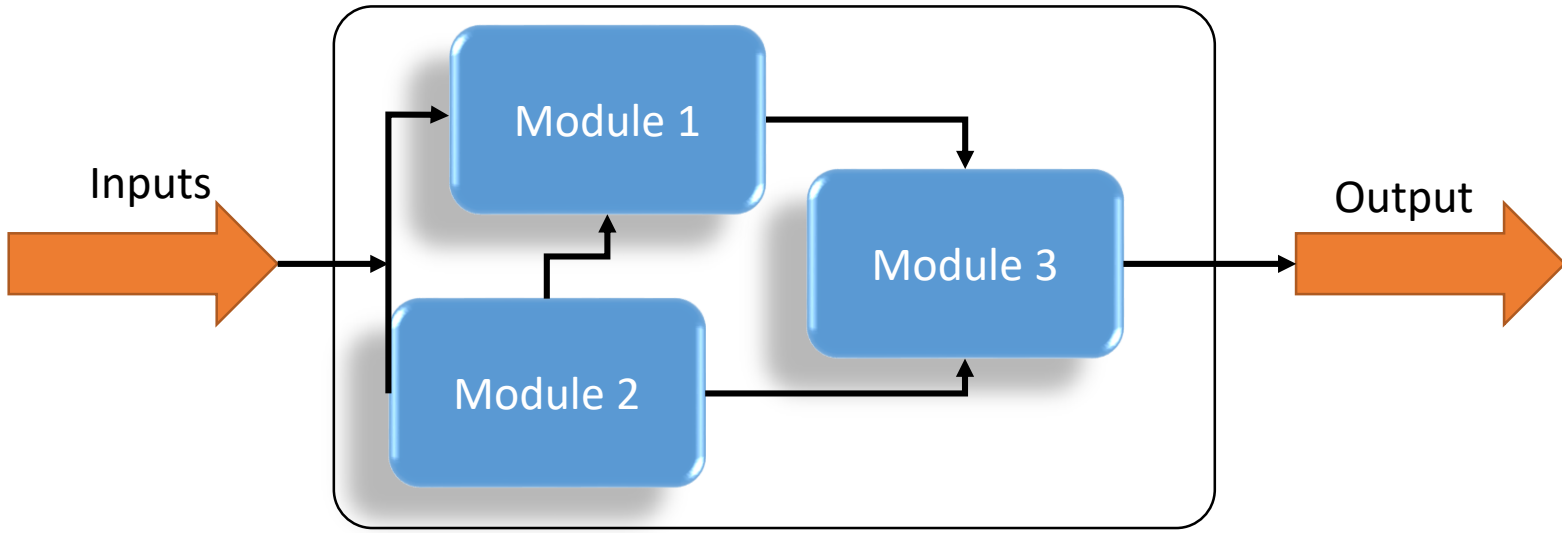
Flot de conception d'un système électronique

La méthodologie Top-down

1ère Phase: On modélise le système sous forme de boite noire. "Je sais ce que ça fait, mais pour le moment je ne sais pas comment."

2ème phase : On partitionne le système en plusieurs parties tout en ayant une vue globale au niveau du système de la manière dont les choses fonctionnent ensemble sans entrer dans les détails de la mise en œuvre.





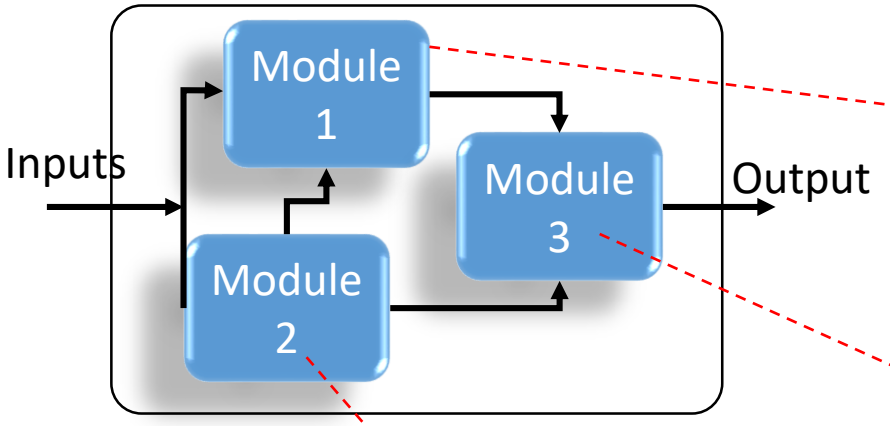
Flot de conception d'un système électronique

La méthodologie Top-down

1ère Phase: On modélise le système sous forme de boite noire. "Je sais ce que ça fait, mais pour le moment je ne sais pas comment."

2ème phase : On partitionne le système en plusieurs parties tout en ayant une vue globale au niveau du système de la manière dont les choses fonctionnent ensemble sans entrer dans les détails de la mise en œuvre.

3ème phase: un code VHDL est développé pour chaque partie du système




```vhdl
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
--use ieee.numeric_std.all;

entity U_Arith is
port (
A: in std_logic_vector (1 downto 0);
B : in std_logic_vector (1 downto 0);
Sel_Operation: in std_logic;
S : out std_logic_vector (1 downto 0));
end entity;

architecture arc of U_Arith is
begin
process (A,B,Sel_Operation) is
begin
case Sel_Operation is
when '0' => S <= unsigned (A) + unsigned (B);
when '1' => S <= unsigned (A) - unsigned (B);
when others => S <= "ZZ";
end case;
end process;
end arc;
```

```vhdl
instantiation: U_Logic
port map (
A => A, B => B, Sel_Operation => Sel_Operation, S => S);

process is
begin
A <= "01";
B <= "01";
Sel_Operation <= '0';
wait for 10 ns;
A <= "01";
B <= "01";
Sel_Operation <= '1';
wait for 10 ns;
A <= "10";
B <= "01";
Sel_Operation <= '0';
wait for 10 ns;
A <= "10";
B <= "01";
Sel_Operation <= '1';
wait for 10 ns;
end process;
end architecture;
```

```vhdl
process is
begin
A <= "01";
B <= "01";
Sel_Operation <= '0';
wait for 10 ns;
A <= "01";
B <= "01";
Sel_Operation <= '1';
wait for 10 ns;
A <= "10";
B <= "01";
Sel_Operation <= '0';
wait for 10 ns;
A <= "10";
B <= "01";
Sel_Operation <= '1';
wait for 10 ns;
end process;
end architecture;
```

Flot de conception d'un système électronique

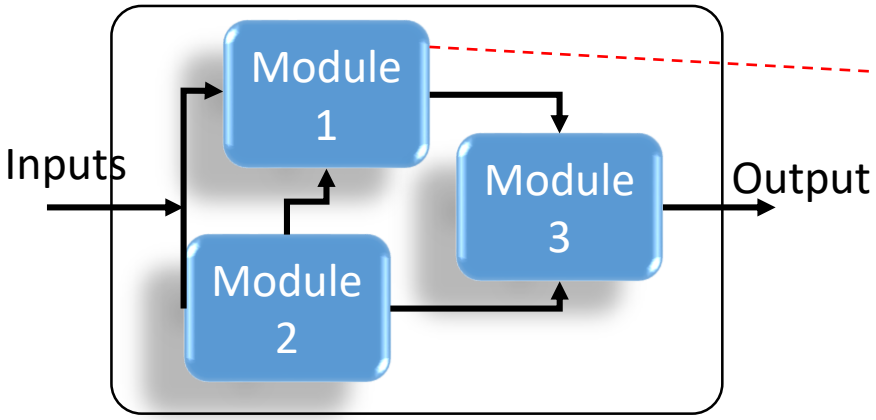
La méthodologie Top-down

1ère Phase: On modélise le système sous forme de boite noire. "Je sais ce que ça fait, mais pour le moment je ne sais pas comment."

2ème phase : On partitionne le système en plusieurs parties tout en ayant une vue globale au niveau du système de la manière dont les choses fonctionnent ensemble sans entrer dans les détails de la mise en œuvre.

3ème phase: un code VHDL est développé pour chaque partie du système

4ème phase: un code VHDL de test est développé pour chaque partie du système pour valider son fonctionnement



Module1.vhd

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.std_logic_arith.all;
4  --use ieee.numeric_std.all;
5
6  entity U_Arith is
7  port (
8  A: in std_logic_vector (1 downto 0);
9  B : in std_logic_vector (1 downto 0);
10 Sel_Operation: in std_logic;
11 S : out std_logic_vector (1 downto 0));
12 end entity;
13
14 architecture arc of U_Arith is
15 begin
16 process (A,B,Sel_Operation) is
17 begin
18 case Sel_Operation is
19 when '0' => S <= unsigned (A) + unsigned (B);
20 when '1' => S <= unsigned (A) - unsigned (B);
21 when others => S <= "ZZ";
22 end case;
23 end process;
24 end arc;
```

Module1_Test.vhd

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.std_logic_arith.all;
4  entity U_Arith_TB is
5  end entity;
6  architecture test_U_Arith of U_Arith_TB is
7  signal A: std_logic_vector (1 downto 0):="00";
8  signal B : std_logic_vector (1 downto 0):="00";
9  signal Sel_Operation: std_logic :='0';
10 signal S : std_logic_vector (1 downto 0);
11
12 component U_Arith is
13 port (
14 A: in std_logic_vector (1 downto 0);
15 B: in std_logic_vector (1 downto 0);
16 Sel_Operation: in  std_logic;
17 S : out std_logic_vector (1 downto 0));
18 end component;
19 begin
20
21 instantiation: U_Arith
22 port map (
23 A => A, B => B, Sel_Operation => Sel_Operation, S => S);
24
```

Flot de conception d'un système électronique

La méthodologie Top-down

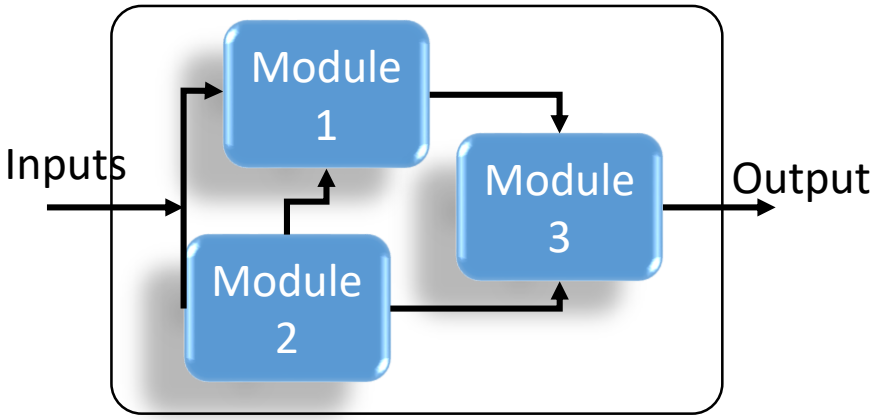
1ère Phase: On modélise le système sous forme de boite noire. "Je sais ce que ça fait, mais pour le moment je ne sais pas comment."

2ème phase : On partitionne le système en plusieurs parties tout en ayant une vue globale au niveau du système de la manière dont les choses fonctionnent ensemble sans entrer dans les détails de la mise en œuvre.

3ème phase: un code VHDL est développé pour chaque partie du système

4ème phase: un code VHDL de test est développé pour chaque partie du système pour valider son fonctionnement

5ème phase: on rassemble tous les modules dans un seul fichier. Puis on test le fonctionnement de l'ensemble



Tous_les_modules.vhd

```
component U_Logic is
port (
A: in std_logic_vector (1 downto 0);
B: in std_logic_vector (1 downto 0);
Sel_Operation: in  std_logic;
S : out std_logic_vector (1 downto 0));
end component;


begin
instantiation1 : U_Arith
port map (A => A1, B => B1, Sel_Operation => Sel_Operation1, S => S_Arith);
instantiation2 : U_Logic
port map (A => A1, B => B1, Sel_Operation => Sel_Operation1, S => S_Logic);

process (A1,B1,Sel_Mode,Sel_Operation1,clk) is
begin
if (clk'event and clk = '1') then
case Sel_Mode is
when '0' => S1 <= S_Arith;
when '1' => S1 <= S_Logic;
when others => S1 <= "ZZ";
end case;
end if;
end process;
end architecture;
```

Test_de_Tous_les_modules.vhd

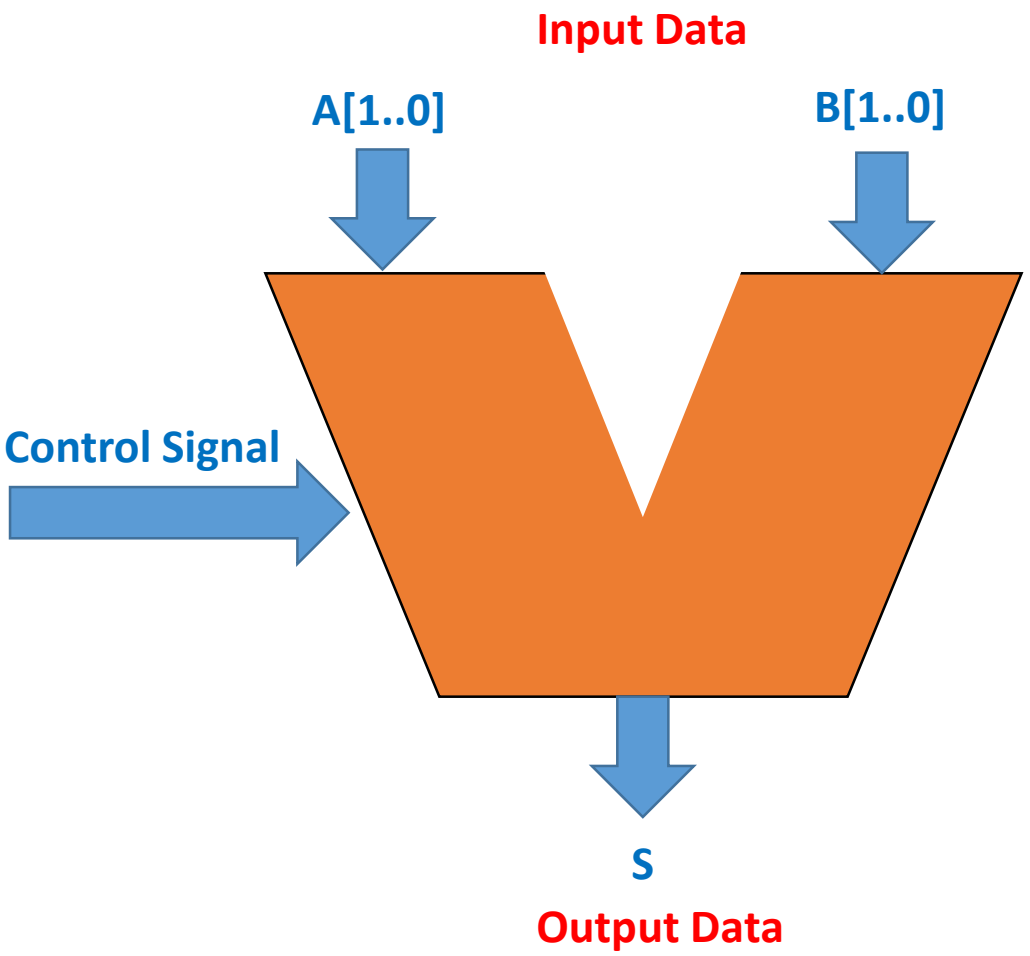
```
instantiation: U_Logic
port map (
A => A, B => B, Sel_Operation => Sel_Operation, S => S);

process is
begin
A <= "01";
B <= "01";
Sel_Operation <= '0';
wait for 10 ns;
A <= "01";
B <= "01";
Sel_Operation <= '1';
wait for 10 ns;
A <= "10";
B <= "01";
Sel_Operation <= '0';
wait for 10 ns;
A <= "10";
B <= "01";
Sel_Operation <= '1';
wait for 10 ns;
end process;
end architecture;
```

Application de la méthodologie TOP-DOWN: conception d'une UAL

- Data:
  - Signal A: bus de 2 bits
  - Signal B: bus de 2 bits

- Signaux de contrôle
  - Mode control:
    - Arithmétique
      - Opérations:
        - Addition
        - soustraction
    - Logique
      - Opérations
        - AND
        - OR
  - Signal d'horloge

- Output
  - Signal S: bus de 2 bits



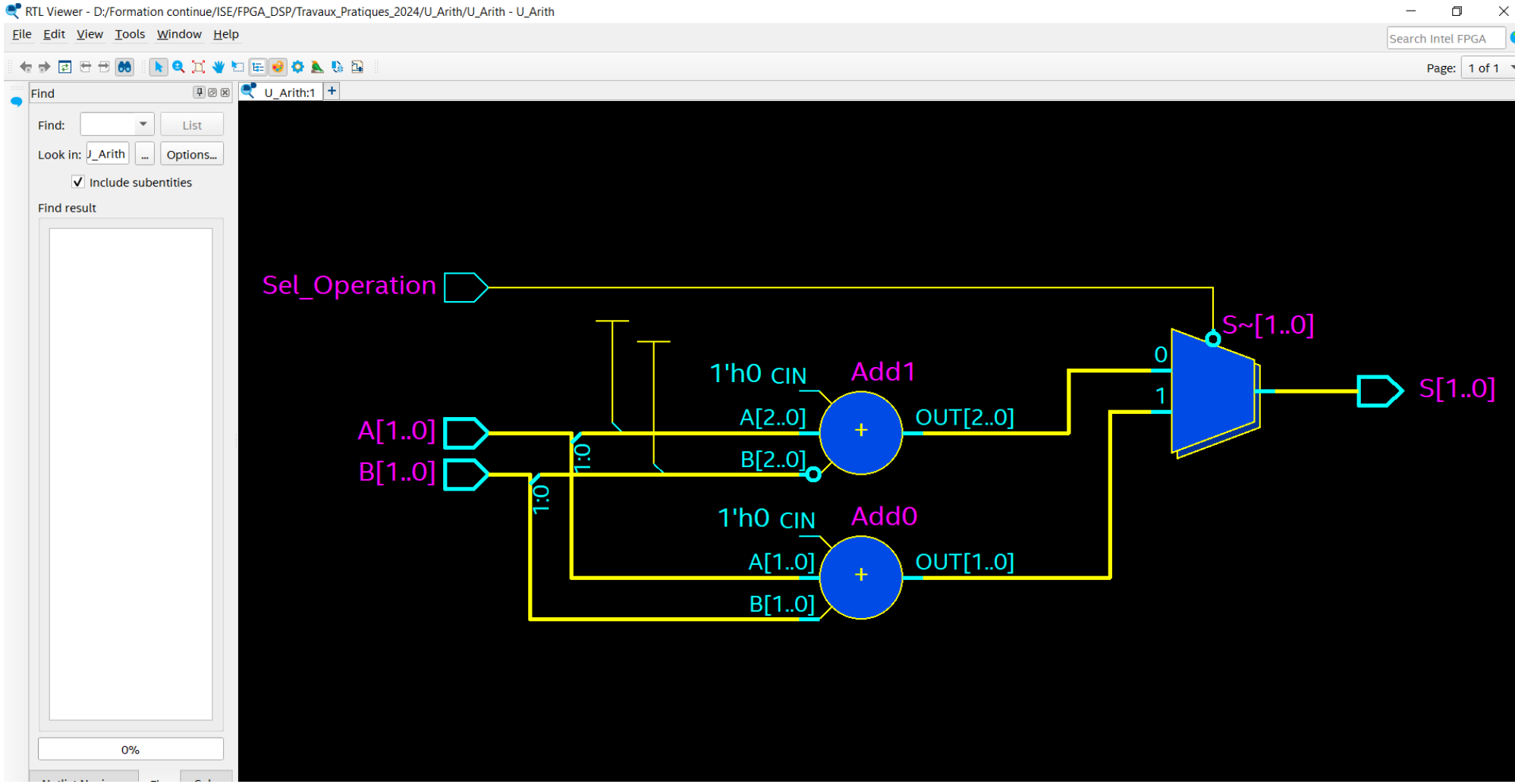
Code décrivant le fonctionnement de l'unite arithmétique

- **Ce premier code permet d'implémenter une unité arithmétique pour effectuer deux opérations arithmétiques à savoir, l'addition et la soustraction**

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.std_logic_arith.all;
4  --use ieee.numeric_std.all;
5
6  entity U_Arith is
7  port (
8  A: in std_logic_vector (1 downto 0);
9  B : in std_logic_vector (1 downto 0);
10 Sel_Operation: in std_logic;
11 S : out std_logic_vector (1 downto 0));
12 end entity;
13
14 architecture arc of U_Arith is
15 begin
16 process (A,B,Sel_Operation) is
17 begin
18 case Sel_Operation is
19 when '0' => S <= unsigned (A) + unsigned (B);
20 when '1' => S <= unsigned (A) - unsigned (B);
21 when others => S <= "ZZ";
22 end case;
23 end process;
24 end arc;
```

La verification au niveau du RTL (Register Transfer Level)

- **L'outil RTL viewer de quartus permet une première vérification de la conception réalisée.**
- **On peut voir clairement que le code permet de synthétiser une unité arithmétique effectuant deux opérations.**
- **Un mux 2 vers 1 est utilisé pour sélectionner l'une des opérations**



Code de test du fonctionnement de l'unite arithmétique (TestBench)

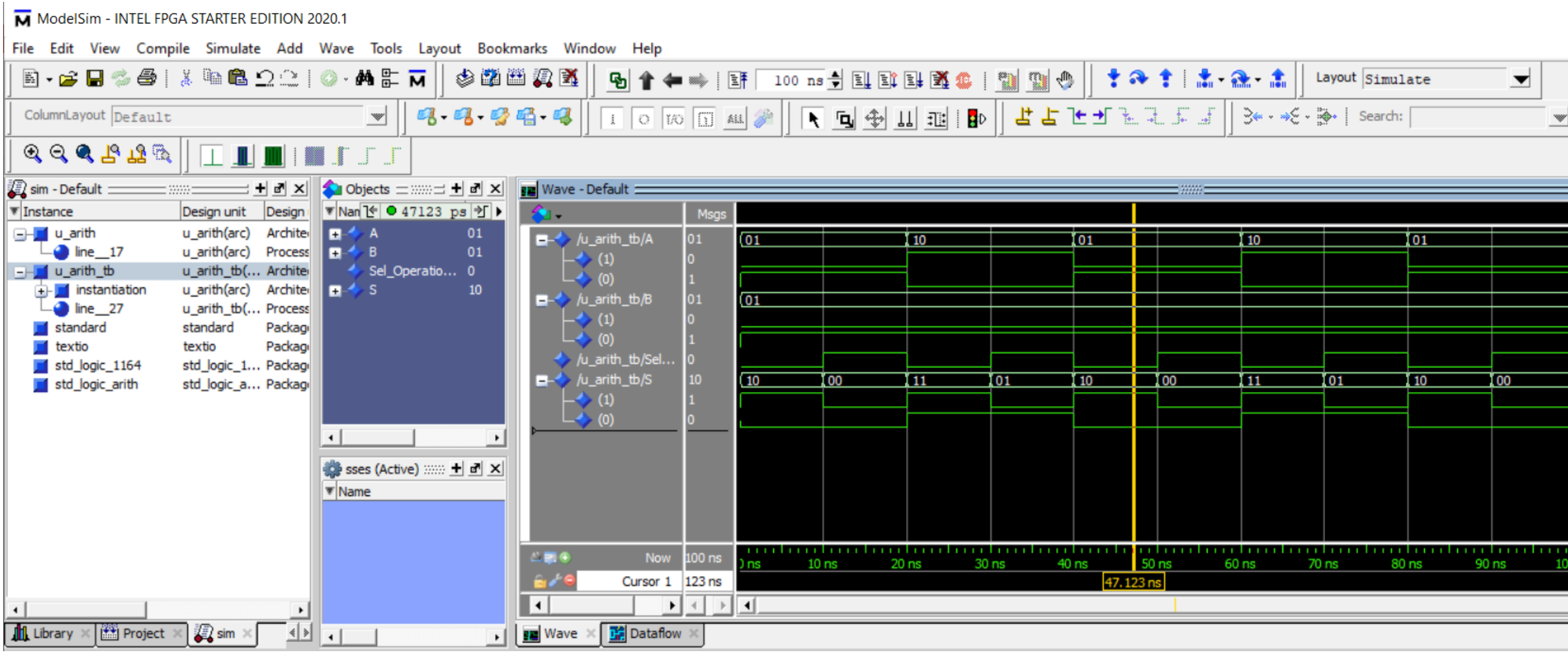
- **Ce deuxième code permet de tester le fonctionnement de l'unité arithmétique réalisée précédemment**

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.std_logic_arith.all;
4  entity U_Arith_TB is
5  end entity;
6  architecture test_U_Arith of U_Arith_TB is
7  signal A: std_logic_vector (1 downto 0):="00";
8  signal B : std_logic_vector (1 downto 0):="00";
9  signal Sel_Operation: std_logic :='0';
10 signal S : std_logic_vector (1 downto 0);
11
12 component U_Arith is
13 port (
14 A: in std_logic_vector (1 downto 0);
15 B: in std_logic_vector (1 downto 0);
16 Sel_Operation: in  std_logic;
17 S : out std_logic_vector (1 downto 0));
18 end component;
19 begin
20
21 instantiation: U_Arith
22 port map (
23 A => A, B => B, Sel_Operation => Sel_Operation, S => S);
24
```

```
25 process is
26 begin
27 A <= "01";
28 B <= "01";
29 Sel_Operation <= '0';
30 wait for 10 ns;
31 A <= "01";
32 B <= "01";
33 Sel_Operation <= '1';
34 wait for 10 ns;
35 A <= "10";
36 B <= "01";
37 Sel_Operation <= '0';
38 wait for 10 ns;
39 A <= "10";
40 B <= "01";
41 Sel_Operation <= '1';
42 wait for 10 ns;
43 end process;
44 end architecture;
```

Résultas de la simulation sous ModelSim

- **Le test dans ModelSim donne les résultats suivants:**



Code décrivant le fonctionnement de l'unité logique

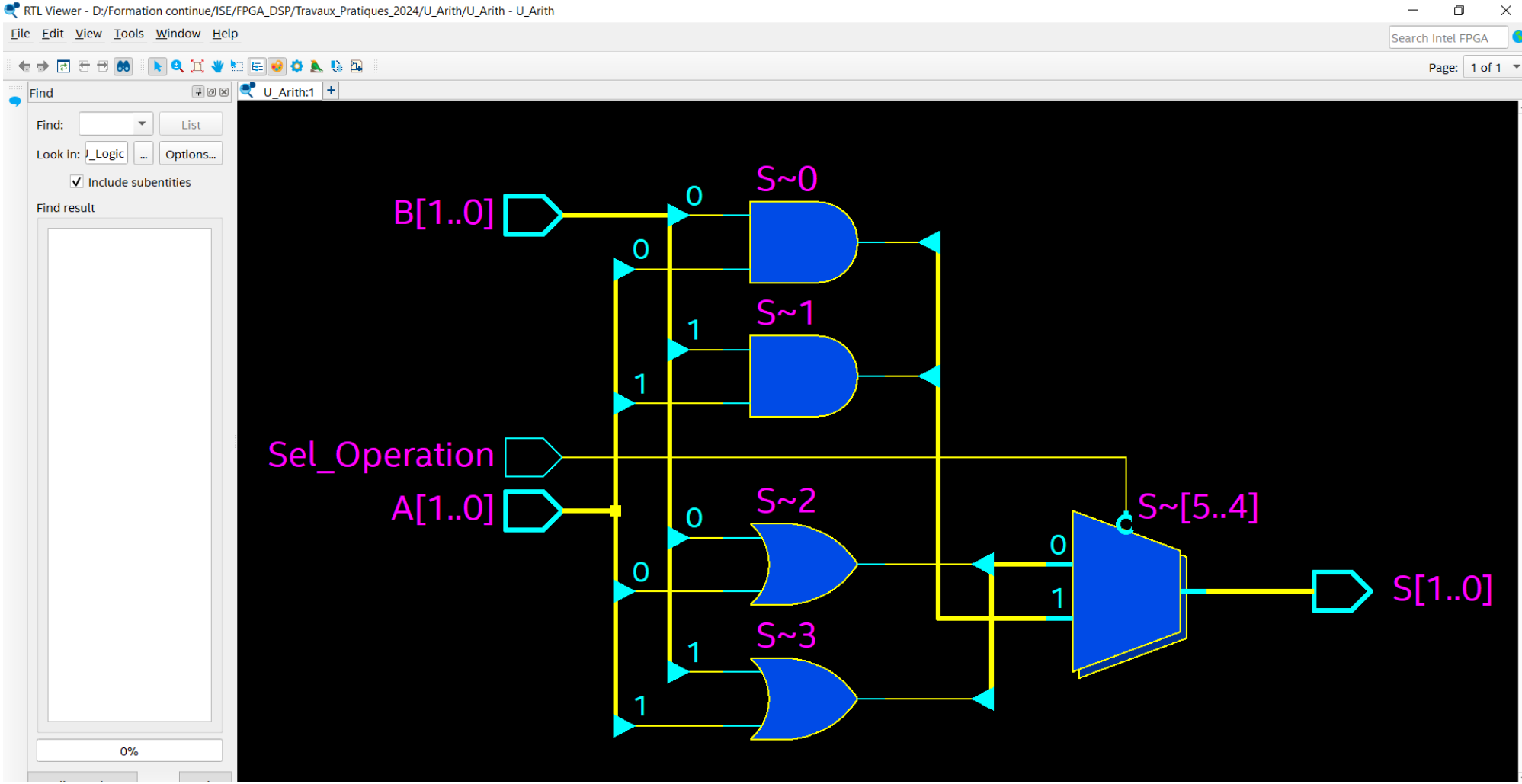
- **Le test dans ModelSim donne les résultats suivants:**

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.std_logic_arith.all;
4  --use ieee.numeric_std.all;
5
6  entity U_Logic is
7  port (
8  A: in std_logic_vector (1 downto 0);
9  B : in std_logic_vector (1 downto 0);
10 Sel_Operation: in std_logic;
11 S : out std_logic_vector (1 downto 0));
12 end entity;
13
14 architecture arc of U_Logic is
15 begin
16 process (A,B,Sel_Operation) is
17 begin
18 case Sel_Operation is
19 when '0' => S <= A and B;
20 when '1' => S <= A or B;
21 when others => S <= "ZZ";
22 end case;
23 end process;
24 end arc;
```

La vérification au niveau du RTL (Register Transfer Level)

- **Le test dans ModelSim donne les résultats suivants:**

- **L'outil RTL viewer de quartus permet une première vérification de la conception réalisée.**
- **On peut voir clairement que le code permet de synthétiser une unité logique effectuant deux opérations.**
- **Un mux 2 vers 1 est utilisé pour sélectionner l'une des opérations**



Code de test du fonctionnement de l'unite logique (TestBench)

- **Ce deuxième code permet de tester le fonctionnement de l'unité logique réalisée précédemment**

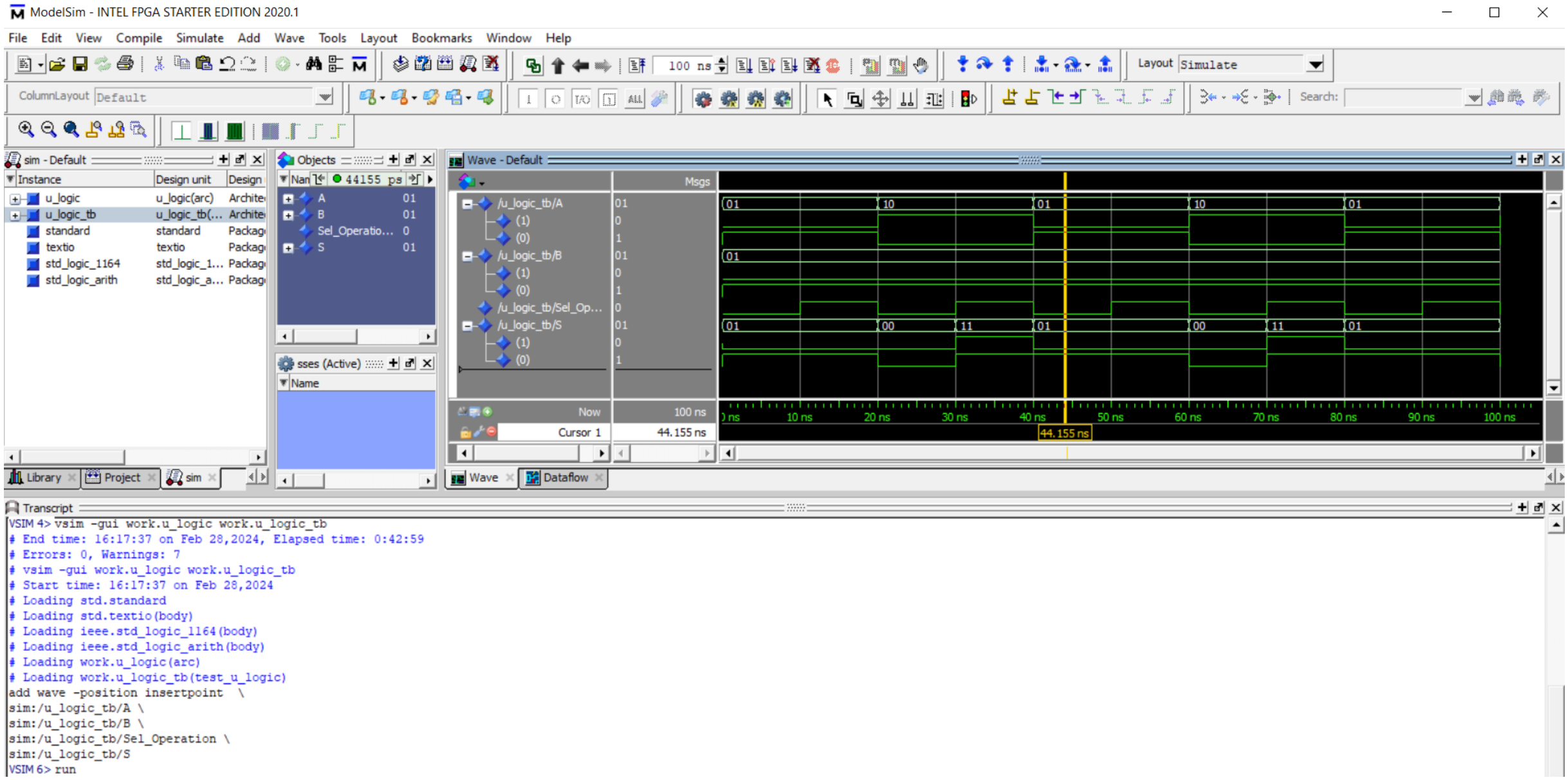
U_Arith.vhd | Compilation Report - U_Arith | U_Arith_TB.vhd | U_Logic.vhd | U_Logic_TB.vhd

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.std_logic_arith.all;
4  entity U_Logic_TB is
5  end entity;
6  architecture Test_U_Logic of U_Logic_TB is
7  signal A: std_logic_vector (1 downto 0):="00";
8  signal B : std_logic_vector (1 downto 0):="00";
9  signal Sel_Operation: std_logic :='0';
10 signal S : std_logic_vector (1 downto 0);
11
12 component U_Logic is
13 port (
14 A: in std_logic_vector (1 downto 0);
15 B: in std_logic_vector (1 downto 0);
16 Sel_Operation: in  std_logic;
17 S : out std_logic_vector (1 downto 0));
18 end component;
19 begin
```

U_Arith.vhd | Compilation Report - U_Arith | U_Arith_TB.vhd | U_Logic.vhd | U_Logic_TB.vhd

```
20
21 instantiation: U_Logic
22 port map (
23 A => A, B => B, Sel_Operation => Sel_Operation, S => S);
24
25 process is
26 begin
27 A <= "01";
28 B <= "01";
29 Sel_Operation <= '0';
30 wait for 10 ns;
31 A <= "01";
32 B <= "01";
33 Sel_Operation <= '1';
34 wait for 10 ns;
35 A <= "10";
36 B <= "01";
37 Sel_Operation <= '0';
38 wait for 10 ns;
39 A <= "10";
40 B <= "01";
41 Sel_Operation <= '1';
42 wait for 10 ns;
43 end process;
44 end architecture;
```

Résultas de la simulation sous ModelSim

- 

Code décrivant le fonctionnement de l'unité arithmétique et logique

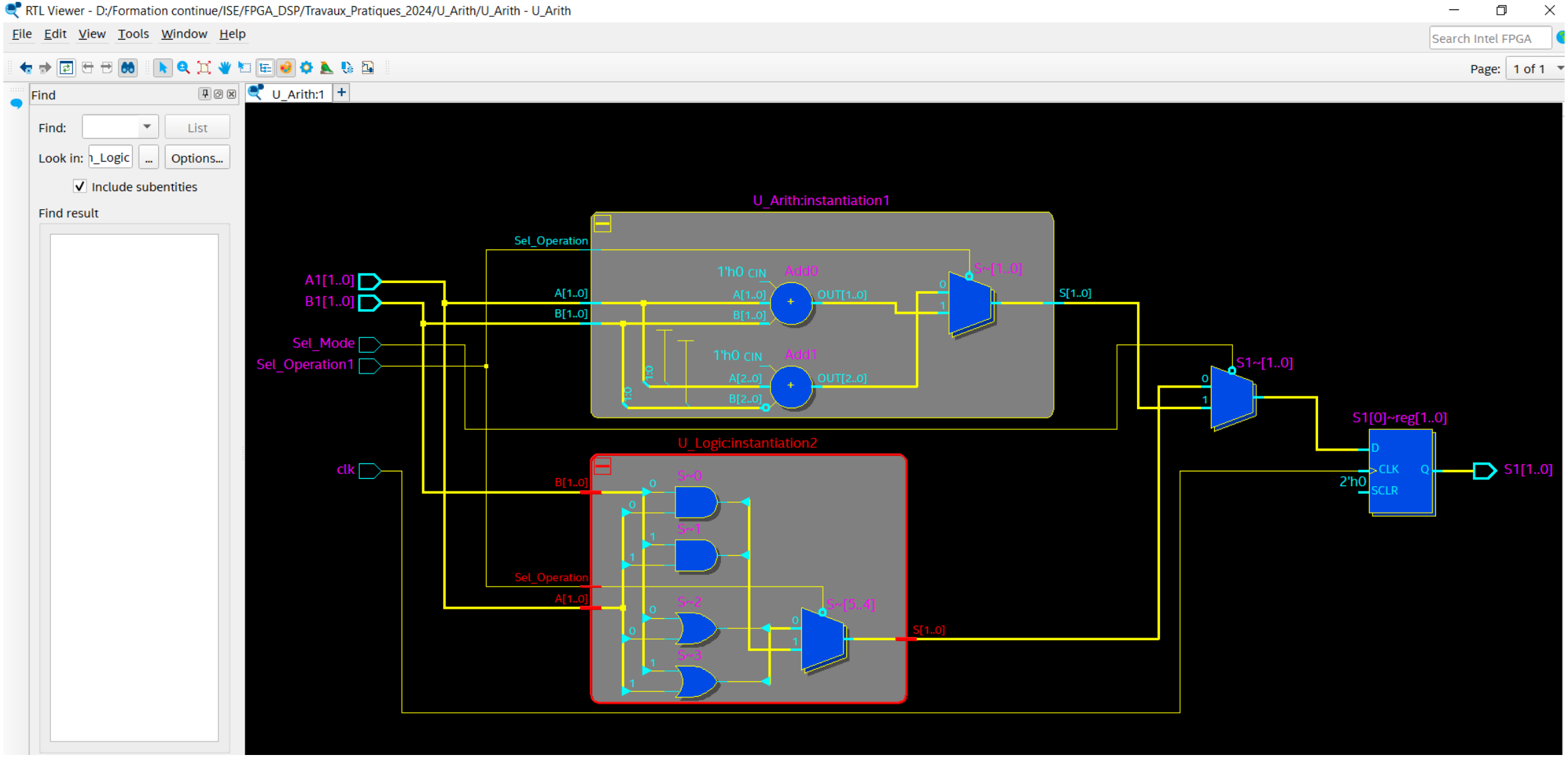
- **Ce code permet de décrire le fonctionnement de l'unité arithmétique et logique**
- **Les deux unités U_Arith et U_Logic ont été instanciées dans ce code pour combiner leurs fonctionnement dans un seul fichier.**

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.std_logic_arith.all;
4  entity U_Arith_Logic is
5  port (
6  A1: in std_logic_vector (1 downto 0);
7  B1 : in std_logic_vector (1 downto 0);
8  Sel_Mode: in std_logic;-- pour choisir U_Arith ou U_logic
9  Sel_Operation1: in std_logic;-- pour choisir l'opération
10 clk : in std_logic;
11 S1 : out std_logic_vector (1 downto 0));
12 end entity;
13
14 architecture bhv of U_Arith_Logic is
15 signal S_Arith : std_logic_vector (1 downto 0);
16 signal S_Logic : std_logic_vector (1 downto 0);
17
18 component U_Arith is
19 port (
20 A: in std_logic_vector (1 downto 0);
21 B: in std_logic_vector (1 downto 0);
22 Sel_Operation: in  std_logic;
23 S : out std_logic_vector (1 downto 0));
24 end component;
25
```

```
27 component U_Logic is
28 port (
29 A: in std_logic_vector (1 downto 0);
30 B: in std_logic_vector (1 downto 0);
31 Sel_Operation: in  std_logic;
32 S : out std_logic_vector (1 downto 0));
33 end component;
34
35
36 begin
37 instantiation1 : U_Arith
38 port map (A => A1, B => B1, Sel_Operation => Sel_Operation1, S => S_Arith);
39 instantiation2 : U_Logic
40 port map (A => A1, B => B1, Sel_Operation => Sel_Operation1, S => S_Logic);
41
42 process (A1,B1,Sel_Mode,Sel_Operation1,clk) is
43 begin
44 if (clk'event and clk = '1') then
45 case Sel_Mode is
46 when '0' => S1 <= S_Arith;
47 when '1' => S1 <= S_Logic;
48 when others => S1 <= "ZZ";
49 end case;
50 end if;
51 end process;
52 end architecture;
```

La vérification au niveau du RTL (Register Transfer Level)

- **On peut voir que le schéma généré correspond parfaitement à l'objectif recherché**



Code décrivant le fonctionnement de l'unité arithmétique et logique

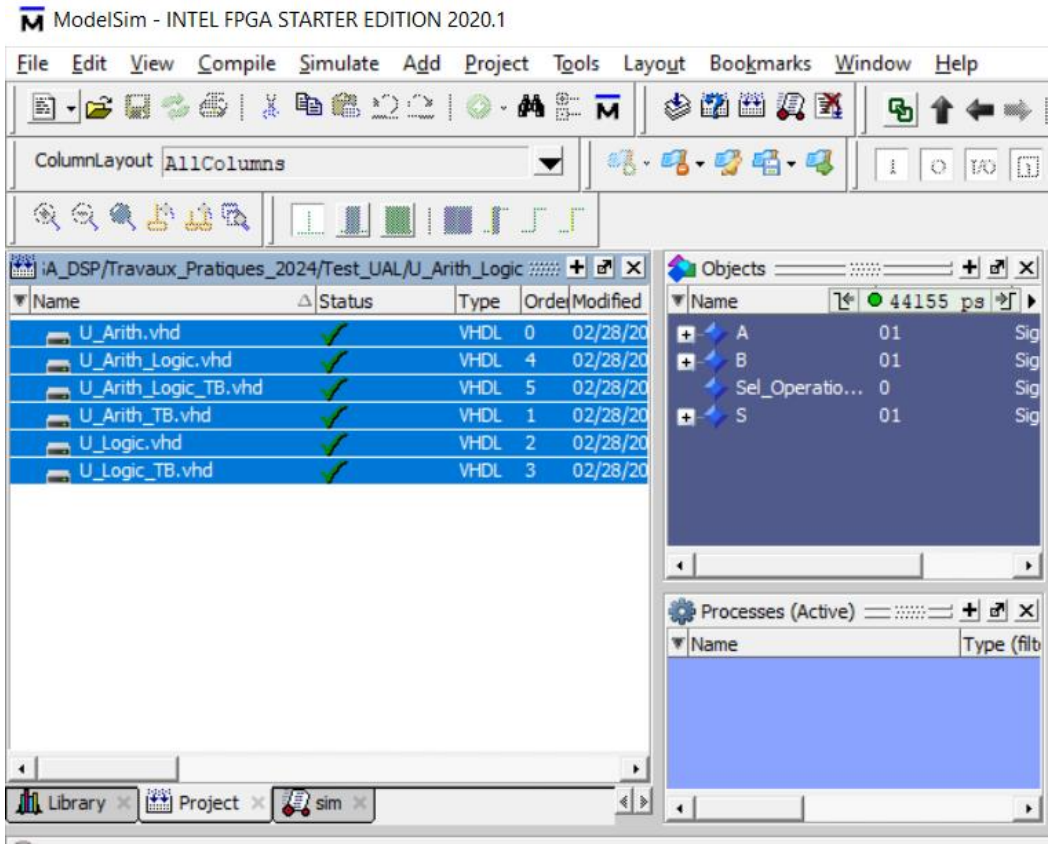
- **Ce deuxième code permet de tester le fonctionnement globale de l'unité arithmétique et logique**

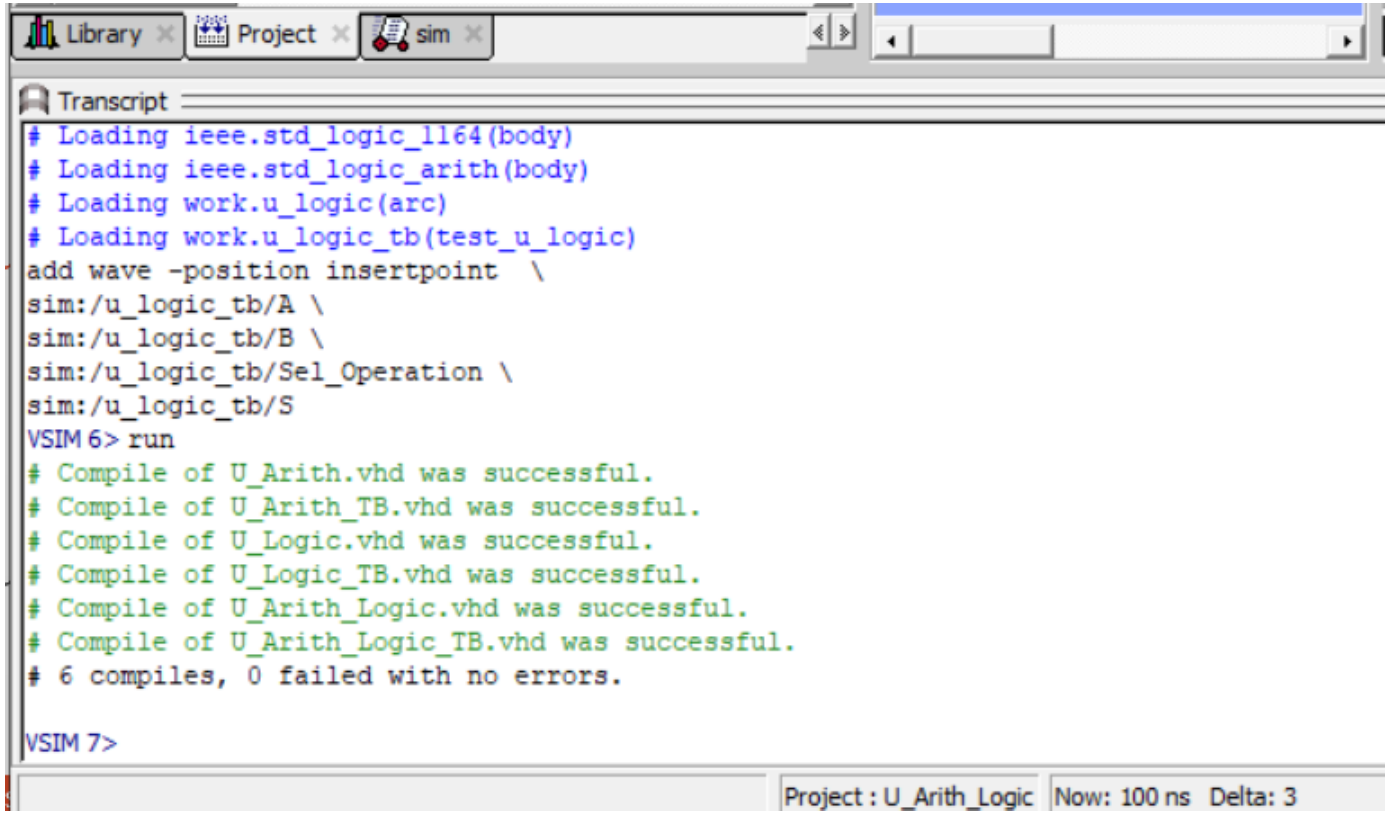
```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.std_logic_arith.all;
4  -------
5  entity U_Arith_Logic_TB is
6  end entity;
7  -----
8  architecture arc of U_Arith_Logic_TB is
9  signal A1:  std_logic_vector (1 downto 0):=(others =>'0');
10 signal B1 :  std_logic_vector (1 downto 0):=(others =>'0');
11 signal Sel_Mode:  std_logic:='0';-- pour choisir U_Arith ou U_logic
12 signal Sel_Operation1:  std_logic:='0';-- pour choisir l'opération
13 signal S1 : std_logic_vector (1 downto 0);
14 signal clk:  std_logic:='0';
15 component U_Arith_Logic is
16 port(
17 A1: in std_logic_vector (1 downto 0);
18 B1 : in std_logic_vector (1 downto 0);
19 Sel_Mode: in std_logic;-- pour choisir U_Arith ou U_logic
20 Sel_Operation1: in std_logic;-- pour choisir l'opération
21 clk:  in std_logic;
22 S1 : out std_logic_vector (1 downto 0));
23 end component;
24
25 begin
```

```
27 instantiation3 : U_Arith_Logic
28 port map (A1 => A1, B1 => B1, Sel_Mode => Sel_Mode,
29  Sel_Operation1 => Sel_Operation1, clk => clk, S1 => S1);
30
31  process is
32  begin
33  Sel_Mode <= '1';
34  Sel_Operation1 <= '0';
35  wait for 40 ns;
36  Sel_Mode <= '0';
37  Sel_Operation1 <= '0';
38  wait for 40 ns;
39  end process;
40
41  process is
42  begin
43  clk <= not (clk);
44  wait for 5 ns;
45  end process;
46
47  process is
48  begin
49  A1 <= "10";
50  B1 <= "01";
51  wait for 10 ns;
52  A1 <= "01";
53  B1 <= "01";
54  wait for 10 ns;
55  A1 <= "00";
56  B1 <= "01";
57  wait for 10 ns;
58 -- A1 <= "10";
73  end process;
74  end architecture;
75
```

Résultas de la simulation sous ModelSim

- **Ce deuxième code permet de tester le fonctionnement de l'unité logique réalisée précédemment**





Résultas de la simulation sous ModelSim

- **Le fonctionnement de l'UAL est bien vérifié comme le montre la figure suivante.**
- **Le résultat, mis dans S1, s'actualise à chaque front montant du signal d'horloge**